SECPI: Searching for Explanations for Clustered Process Instances

Jochen De Weerdt and Seppe vanden Broucke

KU Leuven, Research Centre for Management Informatics (LIRIS)
Naamsestraat 69, B-3000 Leuven, Belgium
jochen.deweerdt@kuleuven.be

Abstract. This paper presents SECPI (Search for Explanations of Clusters of Process Instances), a technique that assists users with understanding a trace clustering solution by finding a minimal set of control-flow characteristics whose absence would prevent a process instance from remaining in its current cluster. As such, the shortcoming of current trace clustering techniques regarding the provision of insight into the computation of a particular partitioning is addressed by learning concise individual rules that clearly explain why a certain instance is part of a cluster.

Keywords: process discovery, trace clustering, user comprehension, instance-level explanations, support vector machines

1 Introduction

Partitioning event logs into multiple groups of process instances is a convenient recipe for addressing the challenge of dealing with complex event logs, i.e. logs presenting a large amount of distinct process behaviour. In the literature, several trace clustering techniques have been described [1–9] that are capable of intelligently splitting up an event log into multiple groups of instances so that process discovery techniques can be applied to subsets of behaviour, with more accurate and comprehensible discovered models as a result. However, the application potential of trace clustering techniques is somewhat hampered by the low level of human comprehension. Concretely, there exist two major problems regarding trace clustering solutions. First of all, it is a non-trivial question to find out what the driving elements are that determine a clustering technique to split up the event log in a particular way. This is because most trace clustering techniques operate at a higher level of abstraction which makes that, for instance, the concept of *distance* between traces is not very insightful as a means to describing a clustering solution. Secondly, end users would like to be able to understand the differentiating characteristics between multiple clusters of process instances, preferably from a *domain perspective*, i.e. relying on control-flow characteristics that are present in the context of the process at hand.

A posteriori comprehension of a clustering solution plays a vital role for the usefulness of separating an event log into multiple subgroups. More specifically, process analysts should be able to understand which factors determine

the delineation of the discovered clusters in order to be able to give an interpretation to the solution. Currently available trace clustering techniques often lack the capability to provide insight into how a certain clustering solution is composed. Therefore, this paper presents a new technique which allows to find explanations that describe which control-flow characteristics of a certain process instance make that this instance pertains to a certain cluster. In the remainder of this paper, it is argued that instance-level explanations can overcome drawbacks of potential alternative explanation techniques, such as for example the visual analysis of the underlying process models. The novel technique, implemented as the *SVMExplainer-plugin* in ProM[1], is inspired by the work of Martens and Provost [10], who put forward an approach for explaining text document classifications. In the context of document classification, one is often confronted with limited comprehensibility of the predictive model, even despite using so-called *white box* techniques such as decision trees or logistic regression, which is mainly due to the high dimensionality. Similarly, such high dimensionality comes into play when characterising process instances by means of binary vectors representing control-flow characteristics.

Against this background, the main contribution of this paper is SECPI (Search for Explanations of Clusters of Process Instances), an algorithm that is capable of finding a minimal set of control-flow characteristics for a process instance, such that if these characteristics were not present, the process instance would not remain within its current cluster. Furthermore, the implementation allows to visualise explanations in the respective process models so that users can easily observe what characteristics make that a process instance belongs to a certain cluster.

2 Trace Clustering

Trace clustering is an interesting approach to deal with the problem that many event logs contain an extensive amount of distinct behaviour (i.e. process variants), because it allows the user to split up a log so that multiple distinct models can be learnt to describe the underlying business process.

2.1 State of the Art

In general, two distinct groups of trace clustering approaches can be discerned with on the one hand techniques that heavily rely on the principle of distance-based clustering, and on the other hand techniques that incorporate a model-driven approach. The first group consists of techniques such as presented in [1, 2, 4, 5, 9], which basically transform an input event log into a propositional format so as to apply well-known clustering techniques from the data mining domain. The technique presented in [4] is slightly different as the similarity between process instances is determined based on string edit operations, while the recently

[1] http://www.promtools.org/prom6/

presented technique in [9] adds a complexity-based procedure to determine the optimal number of clusters based on (approximate) clone detection. The latter group of trace clustering techniques [3, 6, 7] is different in the sense that they are model-driven by relying either on Markov models or Heuristic nets [7]. We refer to the latter paper for a more detailed description and analysis of trace clustering approaches.

2.2 Problem Statement

As indicated in the introduction, the problem with existing trace clustering techniques is that they provide little to no insight into the actual reasoning of partitioning an event log in a particular way. From a model learning perspective, the clustering bias of a trace clustering technique determines how a solution is constructed. Clustering techniques described in the process mining literature employ a wide variety of clustering biases. On the one hand, a subset of techniques relies on the concept of distance as a measure of instance similarity. Model-driven techniques on the other hand rely on maximum likelihood or fitness optimisation. Observe that the ex-post, aggregated fitness of the underlying models is an often employed quality measure for trace clustering solutions, see [4, 7].

For distance-based clustering, typical data mining techniques such as k-means or hierarchical clustering are applied. As such, the distance itself is a potential candidate for explaining a clustering result. For instance, one could visualise the instances in a networked graph or make use of comparative statistical analysis of the underlying variables that determine the inter- and intra-cluster distances. However, a projection of process instances onto process features will typically generate a large amount of variables (e.g. the combined number of 2- and 3-grams for a set of 20 labels is 8 400), which seriously complicates such an approach. To this, it should be added that due to the large amount of variables, distance-based techniques suffer from the curse of dimensionality problem [11]. As described in [12], conventional proximity metrics in high-dimensional space may not be qualitatively meaningful. Therefore, it is argued that the value of the distance concept for assisting users with understanding a trace clustering solution is low. As for model-driven techniques, the natural explanation method is a visual analysis of the resulting cluster models. However, this not only requires a high level of expertise, but is also impacted by the trade-off between recall, precision and generalisation as made by process discovery techniques.

3 Instance-Level Explanations with SECPI

3.1 Approach

This paper describes a completely new analysis approach for explaining the differences between clusters of process instances. The basic idea is shown in Figure 1. *Instead of providing a global explanation, concise if-then rules are learnt for each individual instance*, with a conjunction of control-flow characteristics (e.g.

"sometimes directly follows"-relations) forming the antecedent and the cluster switch as consequence. As such, an explanation is a rule that stipulates which characteristics are the determining factors that make that a certain instance pertains to its current cluster. The goal of our technique is thus to learn accurate yet concise explanations.

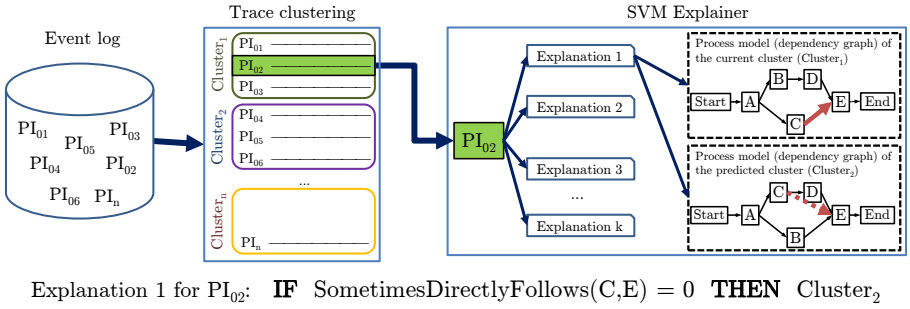

Fig. 1. Overview of SECPI: for each process instance (PI) in the event log, one or more explanations are learnt and ranked according to their length. An explanation is a simple if-then rule with a conjunction of characteristics (as few as possible) which should not be present (i.e. set to zero) in order for the instance to rather belong to a different cluster. The SECPI-plugin in ProM is capable of visually reflecting these key determinants of cluster membership in the respective process models, as illustrated on the right hand side.

Constructing the data set: First, process instances are converted into feature vectors. The implementation supports several attribute templates (e.g. activity presence, always/sometimes weak order relations), however our initial experiments show that the "sometimes directly follows"-attribute template provides solid explanatory power from a control-flow perspective. The *SometimesDirectly Follows*(a, b) attribute for two activities a and b evaluates to true when these two activities both occur in the instance (potentially multiple times) and follow each other directly at least once, and to false otherwise (never follow each other directly or do not both occur). Note that it is out of scope of this study to investigate the optimal configuration of the featurisation step. The data set is completed by adding the appropriate cluster label to each instance. As such, a labeled data set is obtained to which supervised data mining techniques can be applied.

Deriving explanations from a Support Vector Machine (SVM) classifier: As stated earlier, our approach is inspired by [10] in which an algorithm is proposed to find explanations for document classifications. The most important similarity is the use of an SVM-based classifier as the base model from which explanations are derived. As for document classification, SVMs are ideally suited in our context because the use of multiple or complex attribute templates will quickly lead to massive dimensionality. By employing the well-known *liblinear* library for large-scale linear classification based on linear kernel SVMs, our ap-

proach can support data with millions of instances and features. For more details about SVMs, we refer to [13].

The main contribution of this paper consists in adapting the approach in [10] to the context of trace clustering with some key modifications. First, support for multi-class prediction has been developed because in our context it is highly plausible to have more than two clusters. Second, we configure the algorithm in such a way that explanations can be restricted to behaviour present in a process instance (only swaps from 1 to 0 are considered). Third, several performance optimisations have been introduced: we avoid considering attributes with no variability (always 0 or 1), prevent repeat checking of same attribute combinations, and consequently avoid to expand on attribute combinations that have been considered before. These improvements are explained in more detail below.

3.2 Algorithm SECPI

Algorithm 1 provides a formalised overview of the workings of the SECPI algorithm. As inputs, an instance to be explained (a process trace in a cluster) is given, defined as a sequence of binary attributes (generated using the attribute templates as discussed above). Next, a classifier is assumed to be trained over the data set which is able to, for a given feature vector, return a predicted class label and associated score (i.e. probability). Finally, three configuration options have to be set: *iterations* denotes the depth to search for explanations for the given instance. Increasing this value increases the run time but leads to more (albeit longer) explanations. The *zero_to_one* parameter denotes whether 0 to 1 attribute value swaps should be allowed. Since the instance attributes denote characteristics of the instance which *are* present (such as the direct following of two activities, for instance), it is recommended to set this parameter to *False*, as explanations denoting that a trace would not appear in its cluster when it did not present a specific characteristic are generally easier to interpret than explanations denoting that a trace should have a certain characteristic (as the question is then asked where and how exactly this characteristic would manifest itself within the trace). Additionally, since the multitude of all attributes for a trace are set to 0, the list of retrieved rules will be shorter and better fine-tuned to the actual behaviour as seen in the process instance. Finally, *require_support* denotes whether attribute value swaps should be taken into account for attributes which are always set to 0 or 1 (i.e. no variability). Again, it is recommended to set this to a *True* value, as providing explanations which require behaviour which is nowhere seen in the log are most likely less useable than those which do only incorporate seen behaviour.

As output, a set of explanatory rules is returned, formalised as a set of sets of attribute indices. Each set of indices represents a candidate explanation, and should be interpreted as follows: "this process instance would leave its current cluster when all the following attributes would be inverted" – or, in case where *zero_to_one* is set to *False*: "when it would not exhibit the behaviour as represented by these attributes". To construct this set of explanations, the algorithm

Algorithm 1 Formalisation of the SECPI algorithm (as explained in Sect. 4.1)

Input: $I := \langle I_i \in \{0,1\}, i = 1,2,\ldots,|I|\rangle$ % Process instance $I \in$ event log L containing k clusters
Input: $C : L \mapsto \{1,2,\ldots,k\}$ % Trained classifier with scoring function f_C
Input: $iterations := 30$, $zero\_to\_one := False$, $require\_support := True$ % Configuration
Output: Set of explanatory rules R

function SECPI(I, C, $iterations$, $zero\_to\_one$, $require\_support$)
  $c := C(I)$ % Predicted cluster
  $p := f_C(I)$ % Corresponding probability
  $R := \{\}$ % Set of instance explanations (set of sets)
  $E := \{\}$ % Combinations to expand on (set of sets)
  % Search for single attribute explanations
  for all $i := 1 \rightarrow |I|$ **do**
    if $IsAllowedSwap(I, i)$ **then**
      $I' := SwapAttributes(I, \{i\})$
      $c' := C(I')$ % New cluster label
      $p' := f_C(I')$ % New probability
      if $c' \neq c$ **then** $R := R \cup \{i\}$
      else $E := E \cup \{i\}$ **end if**
    end if
  end for
  % Iteratively search for multi attribute explanations
  for all $iteration := 1 \rightarrow iterations$ **do**
    $combo := argmax_{A \in E}(p - f_C(SwapAttributes(\mathrm{I}, \mathrm{A})))$ % Best combination
    $combos' := \{\}$
    for all $i := 1 \rightarrow |I|$ **do** % Expand combination
      $combo' := combo \cup \{i\}$
      if $combo \neq combo' \wedge IsAllowedSwap(I, i) \wedge \neg IsSubsumed(R, combo')$ **then**
        $combos' := combos' \cup \{combo'\}$
      end if
    end for
    for all $combo' \in combos'$ **do**
      $I' := SwapAttributes(I, combo')$
      $c' := C(I')$ % New cluster label
      $p' := f_C(I')$ % New probability
      if $c' \neq c$ **then** $R := R \cup combo'$
      else $E := E \cup combo'$ **end if**
      $E := E \setminus combo$ % Don't check this combination again
    end for
  end for
  return R
end function

function IsSubsumed(R, A)
  % Check whether attributes with indices $\in A$ are subsumed by explanation in R
  for all $E \in R$ **do**
    if $E \in A$ **then return** $True$ **end if**
  end for
  return $False$
end function

function IsAllowedSwap(I, a)
  % Check whether attribute with index a in instance I may be swapped
  $a' := abs(I_a - 1)$
  if $\neg zero\_to\_one \wedge I_a = 0$ **then return** $False$ **end if**
  if $require\_support \wedge \nexists J \in L : J_a = a'$ **then return** $False$ **end if**
  return $True$
end function

function SwapAttributes(I, A)
  % Swap attributes with indices $\in A$ in instance I
  $I' := \langle I'_i \in \{0,1\}, i = 1,2,\ldots,|I| : I'_i = \textbf{if } i \notin A \textbf{ then } I_i \textbf{ else } abs(I_i - 1)\rangle$
  return I'
end function

applies a heuristic, best-first search procedure with pruning. First, each candidate single attribute is evaluated (lines 7 to 15) to see whether rules composed of only one attribute can be found. If swapping an attribute's value does not lead to a class change, a combination of indices (in this case a single index) is added to E to be expanded in the next step.

Next, a number of iterations is performed (lines 17 to 34) as set by the *iterations* parameter. A best-first candidate selection from all currently available combinations to expand on is chosen, based on the classifier's scoring function (line 18). The goal is to first explore the set of attribute indices for which swapping their values moves the instance farthest away from its current class label (i.e. cluster). Expansions on this combination are created by creating a new set of combinations *combos*$'$ by adding each allowed attribute to the set of *combo* (lines 20 to 25). Expansions which are equal to *combo* (i.e. the added attribute was already used in *combo*) or which are subsumed by an already existing explanation (the expansion contains all attribute indices of an existing explanation and thus adds no value) are not considered. Once all expansions are built, they are evaluated to see if they lead to a class change (lines 26 to 33). Expanded combinations are removed from E to prevent them being chosen again in the next iteration (line 32).

As a classification model, we use a combination of k (the number of clusters) SVM models to allow for multi-class classification with SVMs. To retrieve the predicted class label and score, we apply a winner-takes-all strategy as follows. An SVM model is built per cluster to predict whether an instance is in-cluster (label: 1) or out-of-cluster (label: 0). To predict the label and probability of an instance, the probability that the instance is out or in their respective cluster is evaluated for all SVMs (with probability p_k if predicted in-cluster and $1 - p_k$ if predicted out-of-cluster). The SVM model with the highest probability determines the label (and its corresponding probability). Note that other classifiers (such as decision tree or rule based classifiers) could, in theory, also be applied in the SECPI algorithm as long as a scoring function can be defined, and in fact could also return small-sized instance explanations – as is our goal – even though their model itself (in terms of number of rules or decision tree nodes for example) can still be large. However, the construction of such models becomes unwieldy when dealing with high dimensional data sets, so that SVMs remain a better suitable classifier for use within our proposed technique.

4 Conclusion

In this paper, SECPI (Search for Explanations of Clusters of Process Instances), a new technique assisting users with understanding trace clustering results was presented. The need for such a technique stems from the observation that typical trace clustering techniques do not provide sufficient insight into how a clustering solution is composed. In future work, we foresee to expand on a number of closely related topics. First, we plan to inspect the impact of the attribute templates used as they play a crucial role in representing the (control-flow) domain. Also,

we aim at investigating the incorporation of non control-flow-based attributes. Second, aggregation of instance-level explanations is a worthwhile research track as well. The current implementation already supports the investigation of shared explanations amongst groups of instances, which is a preliminary approach to bring our explanation technique to the global level. However, we plan to investigate more intelligent rule clustering and visualisation techniques for this purpose. Finally, we will focus on practical use cases in which SECPI might prove beneficial. User-driven discovery of process model collections from event data is one such area where it can support the feedback mechanism. Furthermore, SECPI is also perfectly capable of relating exogenously defined clusters, e.g. high versus low cost instances, to process-specific control-flow characteristics, a feature often desired in business process improvement cycles.

References

1. Greco, G., Guzzo, A., Pontieri, L., Saccà, D.: Discovering expressive process models by clustering log traces. IEEE Trans. Knowl. Data Eng. **18**(8) (2006) 1010–1027
2. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace clustering in process mining. In Ardagna, D., Mecella, M., Yang, J., eds.: BPM Workshops. Volume 17 of LNBIP., Springer (2008) 109–120
3. Ferreira, D.R., Zacarias, M., Malheiros, M., Ferreira, P.: Approaching process mining with sequence clustering: Experiments and findings. In Alonso, G., Dadam, P., Rosemann, M., eds.: BPM. Volume 4714 of LNCS., Springer (2007) 360–374
4. Bose, R.P.J.C., van der Aalst, W.M.P.: Context aware trace clustering: Towards improving process mining results. In: SDM, SIAM (2009) 401–412
5. Bose, R.P.J.C., van der Aalst, W.M.P.: Trace clustering based on conserved patterns: Towards achieving better process models. In Rinderle-Ma, S. et al., ed.: BPM Workshops. Volume 43 of LNBIP., Springer (2009) 170–181
6. Folino, F., Greco, G., Guzzo, A., Pontieri, L.: Mining usage scenarios in business processes: Outlier-aware discovery and run-time prediction. Data Knowl. Eng. **70**(12) (2011) 1005–1029
7. De Weerdt, J., vanden Broucke, S.K.L.M., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. **25**(12) (2013) 2708–2720
8. Song, M., Yang, H., Siadat, S., Pechenizkiy, M.: A comparative study of dimensionality reduction techniques to enhance trace clustering performances. Expert Systems with Applications **40**(9) (2013) 3722 – 3737
9. Ekanayake, C.C., Dumas, M., García-Bañuelos, L., La Rosa, M.: Slice, mine and dice: Complexity-aware automated discovery of business process models. In Daniel, F., Wang, J., Weber, B., eds.: BPM. Volume 8094 of LNCS., Springer (2013) 49–64
10. Martens, D., Provost, F.: Explaining data-driven document classifications. MISQ **38**(1) (2014) 73–99
11. Bellman, R.E.: Adaptive control processes - A guided tour. Princeton University Press (1961)
12. Aggarwal, C., Hinneburg, A., Keim, D.: On the surprising behavior of distance metrics in high dimensional space. In Bussche, J., Vianu, V., eds.: ICDT. Volume 1973 of LNCS. Springer Berlin Heidelberg (2001) 420–434
13. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov. **2**(2) (1998) 121–167